**Bansilal Ramnath Agarwal Charitable Trust’s**



# Vishwakarma Institute of Technology,Pune-37

*(An Autonomous Institute of Savitribai Phule Pune University)*

**Department of Artificial Intelligence and Data Science**

|  |  |
| --- | --- |
| **Division** | A |
| **Batch** | 3 |
| **GR-no** | 12311951 |
| **Roll no:** | 25 |
| **Name** | Archit Bagad |

LAB ASSIGNMENT

**Write a program to implement: a. Network Routing: Shortest path routing**

Code:

#include <bits/stdc++.h>

using namespace std;

class NetworkRouting

{

protected:

    unordered\_map<char, unordered\_map<char, int>> graph;

public:

    NetworkRouting(const unordered\_map<char, unordered\_map<char, int>> &g) : graph(g) {}

    vector<char> shortestPath(char source, char destination)

    {

        if (graph.find(source) == graph.end() || graph.find(destination) == graph.end())

            return {}; // source or destination not in graph

        unordered\_map<char, int> dist;

        unordered\_map<char, char> prev;

        for (auto &node : graph)

        {

            dist[node.first] = numeric\_limits<int>::max();

        }

        dist[source] = 0;

        priority\_queue<pair<int, char>, vector<pair<int, char>>, greater<>> pq;

        pq.push({0, source});

        while (!pq.empty())

        {

            auto [d, node] = pq.top();

            pq.pop();

            if (node == destination)

                break;

            for (auto &[neighbor, weight] : graph[node])

            {

                int newDist = dist[node] + weight;

                if (newDist < dist[neighbor])

                {

                    dist[neighbor] = newDist;

                    prev[neighbor] = node;

                    pq.push({newDist, neighbor});

                }

            }

        }

        if (dist[destination] == numeric\_limits<int>::max())

            return {}; // No path found

        vector<char> path;

        for (char at = destination; at != source; at = prev[at])

        {

            path.insert(path.begin(), at);

        }

        path.insert(path.begin(), source);

        return path;

    }

};

class AODV : public NetworkRouting

{

    unordered\_map<char, unordered\_map<char, vector<char>>> routeTable;

public:

    AODV(const unordered\_map<char, unordered\_map<char, int>> &g) : NetworkRouting(g) {}

    vector<char> discoverRoute(char source, char destination)

    {

        if (routeTable[source].count(destination) == 0)

        {

            vector<char> path = shortestPath(source, destination);

            if (path.empty())

                return {};

            routeTable[source][destination] = path;

        }

        return routeTable[source][destination];

    }

};

int main()

{

    unordered\_map<char, unordered\_map<char, int>> graph = {

        {'A', {{'B', 1}, {'C', 2}}},

        {'B', {{'A', 1}, {'C', 1}, {'D', 3}}},

        {'C', {{'A', 2}, {'B', 1}, {'D', 1}, {'E', 4}}},

        {'D', {{'B', 3}, {'C', 1}, {'E', 1}}},

        {'E', {{'C', 4}, {'D', 1}}}};

    NetworkRouting nr(graph);

    AODV aodv(graph);

    auto path1 = nr.shortestPath('A', 'E');

    cout << "Shortest path from A to E: ";

    if (path1.empty()){

        cout << "No path found";

    }

    else{

        for (char node : path1)

        cout << node << " ";

    }

    cout << endl;

    auto path2 = aodv.discoverRoute('A', 'E');

    cout << "AODV route from A to E: ";

    if (path2.empty()){

        cout << "No route found";

    }

    else{

        for (char node : path2)

        cout << node << " ";

    }

    cout << endl;

    auto path3 = nr.shortestPath('E', 'B');

    cout << "Shortest path from E to B: ";

    if (path3.empty())

    {

        cout << "No Path Found";

    }

    else

    {

        for (char node : path3)

        {

            cout << node << " ";

        }

    }

    cout << endl;

    return 0;

}

Output:

